Session Objectives

- Practice working with DFA
- Use DFA to model system behaviour
- Understand the relationship between DFA and regular languages
- Apply a phrase structure grammar to generate a language

Exercise

Draw the directed graph that describes the DFA with initial state A, accepting state D and the following transition function.

Input	a	b
Present State		
A	B	D
B	C	D
C	C	C
D	B	D

Exercise II

Which of the following words is accepted by this DFA?

1. $aabb$
2. $bbab$
3. $abbb$
4. $bbaabaa$

Solution II

Which of the following words is accepted by this DFA?

1. $aabb$: No
2. $bbab$: Yes
3. $abbb$: Yes
4. $bbaabaa$: No

Language is generated by the expression $b^*(ab^*)^*$

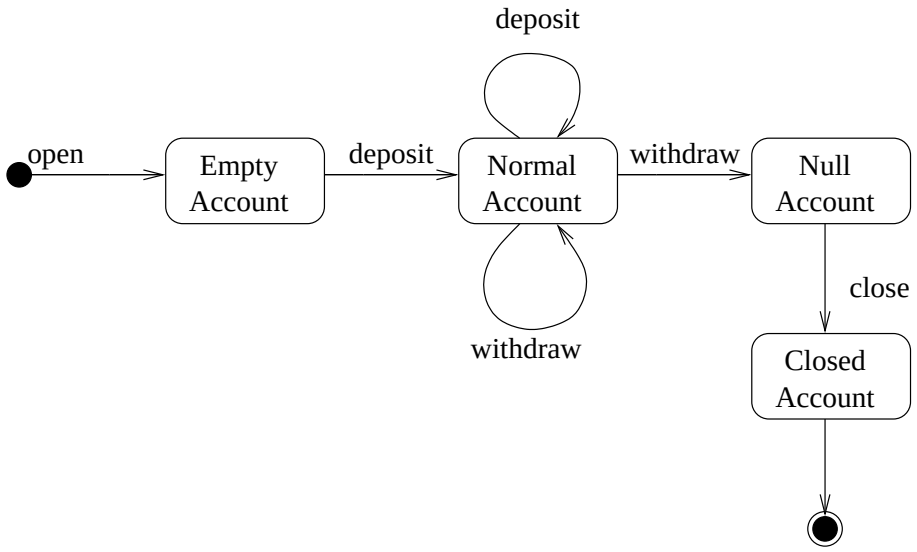
Proof of Non-Regular Languages

- Consider the language whose words all contain a string of 0s followed by an equal number of 1s.
- If this is regular, there is some DFA that accepts it. Suppose that the DFA contains m internal states. Then on reading any string of more than m symbols, the DFA must enter at least one state twice.
- Now consider the string $0^{m+1}1^{m+1}$. This string must be accepted by the DFA, so after reading the last 1 the machine must be in an accepting state. However, when reading the first half of the string, consisting of $m+1$ zeros, the DFA must enter a state (say state S_i) twice.

- What would happen if we inserted another k zeros at this point in the string?
- Since the machine is in state S_i it would still return to S_i: in other words with the string $0^{m+k+1}1^{m+1}$ the machine would end up in the same state as with the string $0^{m+1}1^{m+1}$, which is accepted.
- Thus the DFA would accept a string outside the language, a contradiction.

DFA for System Modelling

- DFA can also be used to model computer system operation, particularly for control systems.
- Here a finite state machine models the way a bank account changes status as it is first opened, used for deposits and withdrawals, all the money is removed and then closed.



Summary

1. A deterministic finite automaton (DFA) consists of a set of states and a transition function that defines the action of the machine when a character is read.
2. DFAs can be represented as directed graphs with loops.
3. A language is accepted by a DFA if and only if it is regular.
4. DFAs can also be used for system modelling.

Phrase Structure Grammars

Because there are useful languages that are not regular, we need to expand the way in which we define a language.

A phrase structure grammar consists of four sets:

Terminal alphabet: the symbols (often represented by lower case letters or numbers) that appear in words (or sentences) of the language.

Non-terminal alphabet: the symbols (often represented by upper case letters) that are used to construct grammatical rules but that do not appear in words.

Start symbol: the symbol that starts the process of generating words.

Productions: the grammar rules used to derive words.

Generating Language

The procedure for generating a grammatical word/sentence is

1. Begin with the start symbol.
2. Use grammar rules to produce new strings.
3. Continue until there are no non-terminals left in the string.

A derivation of a string Y from a string X is a sequence of applications of productions which goes from X to Y. We write $X \Rightarrow Y$.

Example

A language L over the symbols a and b is defined by the following grammar rules, where S is the start symbol.

$$S \to aSb \qquad S \to ab$$

What is the set L?

- We start with the symbol S and use one of the grammar rules to make a replacement. We use rule 1 and get $S \Rightarrow aSb$. Applying the same rule again, we get $S \Rightarrow aSb \Rightarrow aaSbb$.
- Continuing with the same rule, we get $S \Rightarrow a^{n-1}Sb^{n-1}$ after $n-1$ applications. This is not a word in L, since it still contains the non-terminal symbol S. We can remove S by applying rule 2 to obtain $S \Rightarrow a^n b^n$, and so $L = \{a^n b^n \mid n > 0\}$.
- This is equivalent to the language $0^n 1^n$ which we proved was not regular. Hence we conclude that phrase structure grammars can generate new types of language.

Example: Boolean Formulae

The language of Boolean expressions over the set $\{t, f\}$ with the binary operations $\wedge$ and $\vee$ with brackets (and) can be defined with the following grammar rules:

$$S \to (S)B(S) \qquad S \to (S)$$

$$S \to NBN \qquad S \to (S)BN$$

$$S \to NB(S) \qquad S \to N$$

$$N \to f \qquad N \to t$$

$$B \to \wedge \qquad B \to \vee$$

Find all the sentences of length at most four, and show that $t \wedge tf$ is not a sentence.

Solution

For convenience, we summarise the grammar rules using | to represent 'or'. Then we can combine rules with the same left-hand side as follows:

$$S \to (S)BS \mid (S)BN \mid NB(S) \mid (S) \mid NBN \mid N$$

$$N \to f \mid t$$

$$B \to \wedge \mid \vee \tag{1}$$

All of the rules have more symbols on the right than the left, so as we are looking for sentences of up to four terminal symbols, we only need to consider rules with up to four symbols on the right-hand side. This rules out the first three productions from S.

- Both N and B are replaced by terminal symbols only, so it is easy to see that from $S \to N$ we can only derive t and f, of length 1.
- From NBN we can only make the derivations $f \wedge f$, $f \vee f$, $f \wedge t$, $f \vee t$, $t \wedge f$, $t \vee f$, $t \wedge t$, $t \vee t$, all of which have length three.
- Finally, we can derive (S) from S. The only sentences of length at most four that can be derived from (S) use the rule $S \to N$, so we obtain (t) and (f).
- It is easy to see that $t \wedge tf$ is not in the list of valid sentences of length at most four, and hence it is not part of this language.

Exercise

Using the Boolean grammar, show that $(t \wedge t)$ is a grammatical sentence.

Hint: work out a series of productions starting from S that end with this sentence.

Solution

Using the Boolean grammar, show that $(t \wedge t)$ is a grammatical sentence.

$$
\begin{aligned}
S &\rightarrow (S) \\
&\rightarrow (NBN) \\
&\rightarrow (tBN) \\
&\rightarrow (t \wedge N) \\
&\rightarrow (t \wedge t)
\end{aligned}
\qquad\qquad
\begin{aligned}
S &\rightarrow (S) \\
S &\rightarrow NBN \\
N &\rightarrow t \\
B &\rightarrow \wedge \\
N &\rightarrow t
\end{aligned}
$$

Exercise

A language L over the symbols a, b, and c is defined by the following grammar rules, where S is the start symbol and B is a non-terminal symbol.

$$S \to aS$$

$$S \to aB$$

$$B \to bc$$

Find a definition of the set L.

Solution

A language L over the symbols a, b, and c is defined by the following grammar rules, where S is the start symbol and B is a non-terminal symbol.

$$S \to aS$$

$$S \to aB$$

$$B \to bc$$

Find a definition of the set L.

$L = a^*bc$

Session Objectives

- Practice working with DFA
- Use DFA to model system behaviour
- Understand the relationship between DFA and regular languages
- Apply a phrase structure grammar to generate a language